MapMyUni

A

Synopsis Report

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

MUKUL SHIVHARE	**ANVITA GUPTA**	**DEVANSHI NARULA**	**DAKSH BATRA**
R2142221477	R2142221432	R2142221269	R2142220317

under the guidance of

Dr. SONAL TALREJA

School of Computer Science

University of Petroleum & Energy Studies

Bidholi, Via Prem Nagar, Dehradun, Uttarakhand

SEPTEMBER-2024

ABSTRACT

This program is designed to calculate the minimum distance between two locations within a university campus using Dijkstra's algorithm. The application models the campus as a weighted graph where nodes represent key locations (such as buildings, landmarks, and entrances) and edges represent pathways between these locations with weights corresponding to the distances.

Upon receiving the start and end locations from the user, the program applies Dijkstra's algorithm to determine the shortest path in terms of distance. The algorithm efficiently computes the minimum travel distance, ensuring optimal routing for users navigating the campus.

The program features a user-friendly interface that allows for easy input of locations and provides clear, actionable distance results. This tool aids in enhancing campus navigation and optimizing travel routes for students, faculty, and visitors, contributing to a more accessible and organized campus environment.

INTRODUCTION

Navigating a university campus efficiently can be challenging, particularly in large or complex environments where distances between locations can vary significantly. To address this challenge, the development of a program that calculates the minimum distance between two places within a university campus provides a practical solution.

This program leverages Dijkstra's algorithm, a well-established method in computer science for finding the shortest path in a graph with non-negative weights.

In a university setting, the campus can be conceptualized as a weighted graph where nodes represent significant locations such as academic buildings, libraries, dormitories, and parking areas, while edges represent the pathways connecting these locations with associated travel distances.
By employing Dijkstra's algorithm, the program efficiently computes the shortest path between any two given locations, offering users precise distance information and an optimal route.

This tool is designed to enhance the overall campus experience by simplifying navigation, reducing travel time, and improving accessibility.
Whether for new students trying to find their way around, faculty members moving between buildings, or visitors exploring the campus, the program provides a user-friendly interface and accurate distance calculations to facilitate smooth and efficient travel within the university grounds.

PROBLEM STATEMENT

Navigating a university campus can be complex and time-consuming due to its extensive layout and numerous pathways connecting various locations. This complexity is exacerbated for new students, faculty members, and visitors who may be unfamiliar with the campus layout. There is a need for a tool that can accurately calculate the shortest distance between two specific locations within the campus to streamline navigation and enhance overall efficiency.

Since the size of any college/university campus can vary from 30 acres to anywhere around 200 acres, students spend most of their time travelling between different buildings. New students/faculty/parents feel it is inconvenient to search their way inside the campus. Therefore, a navigation system is required to find the optimal path within the campus and for the problem.

The core problem addressed by this project is the lack of an efficient and accurate method for determining the minimum travel distance between any two points within a university campus. To address this issue, the project involves developing a program that utilizes Dijkstra's algorithm to compute the optimum path in a weighted graph representation of the campus.

This program will:

1. Model the campus as a graph with nodes representing key locations and edges representing pathways with distances.
2. Implement Dijkstra's algorithm to find the minimum distance between two user-specified locations.
3. Provide an intuitive user interface for easy input and output of location data.

The goal is to create a tool that simplifies campus navigation, reduces travel time, and provides accurate and efficient routing information, ultimately enhancing the user experience for all campus users**.**

ALGORITHM

Dijkstra's algorithm is a classical algorithm used for finding the shortest path between nodes in a graph, which may represent, for example, road networks or network topologies. It is particularly useful for graphs with non-negative edge weights, such as distances between locations on a university campus.

Here's a step-by-step explanation of how Dijkstra's algorithm works:

1. Initialization:

    - *Start Node*: Begin with the source node (the starting location). Set its distance to zero because the distance from the source to itself is zero.
    - *Distance Table*: Initialize a distance table where each node's distance is set to infinity (∞), except for the source node, which is set to zero.
    - *Priority Queue*: Use a priority queue (or a min-heap) to keep track of nodes to be processed. Initialize this queue with the source node.

2. Processing Nodes:

    - *Extract-Min*: Remove the node with the smallest distance value from the priority queue. This node is considered to have the shortest known distance at this point.
    - *Update Distances*: For each neighbor of this node, calculate the tentative distance from the source node. The tentative distance is the sum of the distance to the current node and the weight of the edge connecting the current node to the neighbor.
    - *Check and Update*: If the calculated tentative distance is less than the known distance for the neighbor node, update the distance table with the new shorter distance and add the neighbor to the priority queue with this updated distance.

3. Termination:

- *Repeat*: Continue the process of extracting the node with the smallest distance and updating distances for its neighbors until all nodes have been processed or the priority queue is empty.
- *Shortest Path*: Once all nodes have been processed, the distance table will contain the shortest distance from the source node to each node in the graph.

SYSTEM REQUIREMENTS

Software Requirements

Operating System	:	Windows 11/10/8/7 (32-bit or 64-bit)/ Linux
Software	:	Text Editor/VSCode/JVM
Compiler	:	JDK

Hardware Requirements

Processor	:	Dual Core 2.7 GHz or better
RAM	:	4 GB or higher

DATA

To implement the project of calculating the minimum distance between two places on a university campus, appropriate data and data structures must be utilized to represent the campus layout and facilitate efficient computation using Dijkstra's algorithm. Below is a description of the sample data and the native data structures suitable for the project.
Considering the UPES BIDHOLI campus with the following locations and pathways:

Locations (Nodes):

1. A: Main Entrance

- B: MAC
- C: Library
- D: Design Block
- E: Frisco
- F: Gandhi Chowk
- G: Computer Science Blocks
- H: Health Sciences Blocks
- I: Science Laboratories Block
- J: Infirmary
- K: Boys Hostel
- L: Football Arena & Girls Hostel
- M: Food Court
- N: Hubble(Administration Block)
- O : Girls Hostel

Pathways (Edges) with Distances:

- A to B: 150 meters
- A to C: 292 meters
- B to C: 145 meters
- B to D: 110 meters
- C to D: 40 meters
- D to E: 134 meters
- C to E: 145 meters
- E to G: 145 meters
- F to H: 141 meters
- F to L: 163 meters
- A to F: 310 meters; 275 meters
- A to I: 445 meters; 410 meters
- B to L: 324 meters
- A to M: 156 meters
- F to O: 215 meters
- F to J: 115 meters
- F to K: 135 meters
- M to F:108 meters
- A to G: 400 meters; 360 meters
- A to E: 326 meters; 214 meters;220 meters

DATA STRUCTURES

1. Graph Representation:

    - *Adjacency List*: This is the most suitable data structure for representing the campus graph. Each node will have a list of edges with associated weights. The adjacency list is efficient for sparse graphs, like the campus network, where not all nodes are directly connected.

2. Priority Queue:

    - *Min-Heap*: A priority queue (min-heap) is used to efficiently retrieve the node with the smallest tentative distance during the execution of Dijkstra's algorithm

3. Distance Table:

    - *Dictionary*: A dictionary is used to maintain the shortest known distances from the source node to all other nodes. This allows quick lookups and updates.

SWOT ANALYSIS

A SWOT analysis evaluates *the Strengths, Weaknesses, Opportunities, and Threats* related to the development and implementation of a campus navigation tool based on Dijkstra's algorithm.
This analysis compares the proposed tool with existing solutions to highlight its advantages and identify areas for improvement.

Strengths:

1. Accurate Shortest Path Calculation:

    - *Advantage*: Dijkstra's algorithm is well-suited for finding the shortest path in graphs with non-negative weights, ensuring accurate distance calculations.

- *Comparison*: Unlike heuristic-based methods or simple Euclidean distance approaches, Dijkstra's algorithm guarantees optimal solutions for shortest path problems.

2. Efficiency:

   - *Advantage*: With a time complexity of **O((V+E)logV),** the algorithm is efficient for handling graphs with a moderate number of nodes and edges.
   - *Comparison*: Compared to brute-force methods, which have exponential complexity, Dijkstra's algorithm provides a significant performance advantage in real-time applications.

3. User-Friendly Interface:

   - *Advantage*: The program can feature an intuitive interface for easy input and navigation, making it accessible to users with varying levels of technical expertise.
   - *Comparison*: Existing solutions such as paper maps or manual directions lack interactive and dynamic features, making them less user-friendly.

4. Real-Time Updates:

   - *Advantage*: The program can be updated to reflect changes in the campus layout or new pathways, ensuring the information remains current.
   - *Comparison*: Static maps or traditional navigation aids do not offer real-time updates, potentially leading to outdated information.

Weaknesses:

1. Setup Complexity:

   - *Disadvantage*: Creating and maintaining an accurate graph of the campus with all pathways and distances is a valuable investment of time that will provide lasting benefits.

- *Comparison*: Some existing campus maps or GPS-based systems may already have data available, allowing for a quicker setup while still offering valuable insights.

2. Scalability:

   - *Disadvantage*: As the campus grows or becomes more complex, the graph may become large and require more computational resources.
   - *Comparison*: Simple solutions or less detailed maps may perform better in smaller or less complex environments.

3. Dependency on Accurate Data:

   - *Disadvantage*: The accuracy of distance calculations depends on the precision of the input data. Errors in data collection can lead to incorrect results.
   - *Comparison*: Some existing systems might use more robust data sources or incorporate error-checking mechanisms.

4. User Dependency on Technology:

   - *Disadvantage*: The tool requires users to have access to a device with the program installed, which may not always be feasible.
   - *Comparison*: Traditional navigation methods like printed maps do not rely on technology, making them universally accessible.

Opportunities:

1. Integration with Other Systems:

   - *Opportunity*: The tool can be integrated with other campus systems such as event schedules, building directories, and emergency alerts.
   - *Comparison*: Unlike standalone systems, integrated solutions offer enhanced functionality and user experience.

2. Enhanced Features:

    - *Opportunity*: Future versions could include features such as voice navigation, accessibility options for disabled users, and integration with mobile apps.
    - *Comparison*: Existing solutions may lack these advanced features, giving the tool a competitive edge.

3. Improved User Engagement:

    - *Opportunity*: Interactive maps can enhance user engagement and satisfaction, leading to better campus navigation.
    - *Comparison*: Less interactive systems might not engage users as effectively.

4. Customization for Different Campuses:

    - *Opportunity*: The tool can be customized and adapted for different university campuses, making it a versatile solution.
    - *Comparison*: Generic navigation systems may not provide the level of customization needed for specific campus layouts.

Threats:

1. Technological Challenges:

    - *Threat*: Technical issues such as software bugs, data corruption, or device compatibility problems could affect the reliability of the tool.
    - *Comparison*: Traditional methods are less prone to technical failures but may lack advanced functionality.

2. User Adoption:

    - *Threat*: Users may be resistant to adopting new technology or may prefer existing methods, limiting the tool's effectiveness.
    - *Comparison*: Established methods might have an ingrained user base, making it challenging to transition to a new tool.

3. Data Security:
    - *Threat*: Handling user data and campus layout information poses a risk of data breaches or privacy concerns.
    - *Comparison*: Systems that are not connected to external networks may be less vulnerable to data security threats.

4. Cost of Development and Maintenance:
    - *Threat*: The costs associated with developing, deploying, and maintaining the tool could be significant.
    - *Comparison*: Existing systems with lower initial investment might be more cost-effective, though they may lack advanced features.

AREA OF APPLICATION

MapMyUni will help newcomers to the college campus to navigate and find the shortest paths to various locations, such as classrooms, libraries, cafeterias, and administrative offices.
It will help students save time, reduce the chances of getting lost, and provide an easy way to explore the campus.

REFERENCES

1. THE BENEFITS OF INTERACTIVE CAMPUS MAPS by Sofia Spagnuolo, (23 June 2021) - link

2. Coimbatore Institute of Technology Campus Navigation System (Version 1.0) by Sheryl Sharon G, Rohit Vikaas P, Chanduru A, Barathkumar S, Harsha Vardhan P, Dr. M. Mohanapriya(October 2023) - link

3. Distances between source node to all other nodes by FelixTechTips
   Link: https://youtu.be/bZkzH5x0SKU?si=OEFkadszJPwpS07L

4. Shortest Path by CodeHelp-by Babbar
   Link: https://youtu.be/abIEXKFpLNE?si=EFWawlqm3YAJvkCm

5. Graph Traversal & Graph Traversal Algorithms by CODEwithHarry
Link:https://youtube.com/playlist?list=PLUPfVLTCq5c2SmJeiouZ53mrkL46BvmI8&si=ZV58QbPp3tni6Eco

6. Searching in Binary Search Tree by CODEwithHarry:
Link: https://youtu.be/XaXsJJh-Q5Y?si=vQVBdOCTcCJoa6R7

7. Priority Queue by WilliamFiset
Link: https://youtu.be/wptevk0bshY?si=oJh70gkE059-CT9f

8. SMART CAMPUS NAVIGATION SYSTEM by Naik Amisha, Nese Likhita, Sharma Anupama, Tripathi Richa, Swapna Patil - December 2022, Conference: IJCRT at: MUMBAI - link

9. AN EXPERIMENT ON THE PERFORMANCE OF SHORTEST PATH ALGORITHM by Simon Chan Yew Meng, Nur'ayuni Adnan, Syazwan Syafiqah Sukri, and Wan Mohd Nazmee Wan Zainon, Knowledge Management International Conference (KMICe) 2016, 29 – 30 August 2016, Chiang Mai, Thailand - link

10. The distances used as data are as per GoogleEarth - link